A Force-Directed Method for Large Crossing Angle Graph Drawing

Peter Eades* University of Sydney
Weidong Huang† University of Sydney
Seok-Hee Hong‡ University of Sydney

Abstract

Recent empirical research has indicated that human graph reading performance improves when crossing angles increase. However, *crossing angle* has not been used as an aesthetic criterion for graph drawing algorithms so far. In this paper, we introduce a force-directed method that aims to construct graph drawings with large crossing angles. Experiments indicate that our method significantly increases crossing angles. Surprisingly, the experimental results further demonstrate that the resulting drawings produced by our method have fewer edge crossings, a shorter total edge length and more uniform edge lengths, compared to classical spring algorithms.

Keywords: Graph visualization, graph drawing, crossing angle, cosine force, force-directed method.

Index Terms: G.2.2 [Discrete Mathematics]: Graph Theory—Graph Algorithms

1 Introduction

A great deal of real world data have a relational structure and can be modeled as graphs. Graphs are usually drawn as node-link diagrams so that humans can make sense of the underlying structure. The past two decades have seen a fast growing body of research dedicated to designing algorithms to construct aesthetically pleasing drawings of graphs [6]. While judgement of the quality of a drawing is subjective, a number of aesthetic criteria are generally accepted, including the following:

- Small number of edge crossings
- Even distribution of vertices
- Uniform edge length
- Small drawing area
- Maximum angular resolution

These criteria were originally proposed based on human intuition. However, some have been validated in user studies, mainly conducted by Purchase et al. (e.g., [27]), indicating that drawings satisfying such criteria yield some insights from the end user's point of view. For example, edge crossings were found to have the greatest impact on human graph reading performance. Recently, researchers have begun investigating theories from general psychology and adopting empirical methods in order to develop human-centered criteria for graph drawing [30, 31]. Among them, Ware et al. [31] studied results from neurophysiology and suggested that edges that cross at nearly 90 degrees are less likely to be confusing than those crossing at acute angles (see Figure 1). The effect of crossing angle was subsequently observed in a qualitative eye

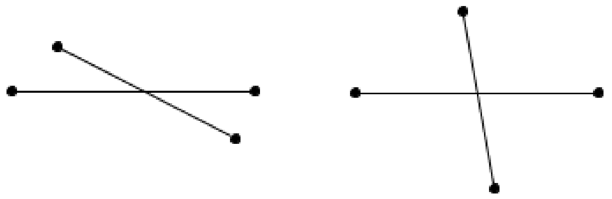


Figure 1: Theoretical and empirical research reveals that the crossing on the left is more confusing than that on the right.

tracking study by Huang [19]. This was validated quantitatively in a controlled experiment [20]. To be more specific, it was found that human graph reading performance improves when crossing angle increases.

While a great deal of attention has been focused on reducing the number of crossings, little research has been done on how to handle the remaining crossings. Didimo et al. [5] initiated a study of combinatorial questions related to drawing graphs with right angle crossings. Dunne and Shneiderman [7] list crossing angle as a "readability metric". However, in this previous research, no algorithm for producing graph drawings with large crossing angles has been proposed. In this paper, we introduce a force-directed method that aims to construct graph drawings with large crossing angles, called BIGCROSS. Experiments indicate that our method significantly increases crossing angles. Surprisingly, our experimental results further demonstrate that the resulting drawings produced by our method also have fewer edge crossings, a shorter total edge length and more uniform edge lengths, compared to a classical spring algorithm.

2 The Classical Spring Algorithm

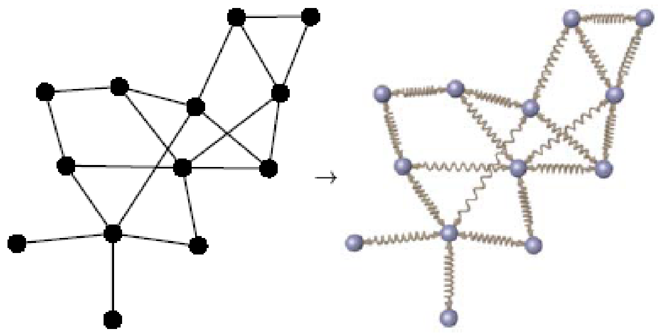


Figure 2: The spring embedder model (taken from Brandes [3])

Force-directed graph drawing has been studied extensively (e.g., [8, 12]). In the *classical spring algorithm*, a graph is modeled as a physical system, in which vertices are replaced with steel rings, and edges are replaced with rings (see Figure 2). Springs pull connected vertices toward to each other when stretched, while they push vertices apart when compressed, following Hooke's law:

$$f_s = k_s(d - l). \quad (1)$$

*e-mail: peter@it.usyd.edu.au
†e-mail: whua5569@it.usyd.edu.au
‡e-mail: shhong@it.usyd.edu.au

The repulsive force between all rings follows an inverse square law:

$$f_r = k_r/d^2. \tag{2}$$

Here k_s and k_r are constants, d is the Euclidean distance between vertices and l is the natural length of the spring.

Starting with an arbitrary placement of vertices, the algorithm calculates the combined force on each vertex and moves the vertices accordingly. This process is repeated for a fixed number of times and the resulting pictures are usually aesthetically pleasing.

In the last two decades, the spring embedder model has been refined and extended in many different ways. Gansner and North [13] introduced two post-processing techniques to avoid vertex-vertex and vertex-edge overlaps. Brandes and Wagner [4] defined a random field model for drawing so-called *train graphs* to avoid overlaps and small angles between edges resulting from straight-line edges. Lin and Yen [26] introduced a repulsive force between adjacent edges to improve angular resolution. To visually separate clusters, Huang et al. [18] introduced dummy vertices representing individual clusters and repulsive forces that act between the dummy vertices. Recently, Hu and Koren [22] proposed and implemented two post-processing algorithms to overcome warping effects resulted from the spring embedder model. Holten and van Wijk [16] introduced a self-organizing approach in which edges are modeled as flexible springs that can attract each other for edge bundling.

More sophisticated techniques have been proposed for various purposes (e.g., [17, 21, 23, 24]). For excellent reviews, see [3, 6].

3 The BIGCROSS Method

The beauty of force-directed graph drawing is that we can simply add up energy functions, each of which aims to maximize a specific aesthetic criterion. Our new "BIGCROSS" method, defined below, is an extension of the classical spring algorithm. We define a new force that increases the angle between crossing edges, and apply it in addition to the forces of the classical model.

3.1 BIGCROSS Force Magnitude

The BIGCROSS algorithm extends to introduce an extra force called the *cosine force* to increase crossing angle. The basic idea on how the cosine force works is as follows.

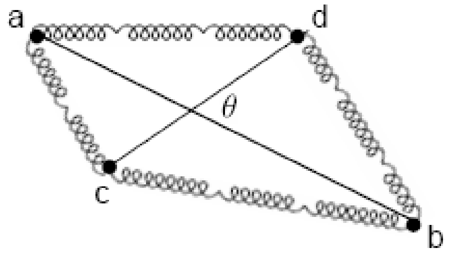


Figure 3: The model of cosine force

As shown in Figure 3, if two edges (a, b) and (c, d) cross, then their endpoints a, b, c and d will be connected by special springs. These springs are special in that they work together and apply a non-Hooke's-law force on each of the endpoints, in such a way that when the crossing angle increases, energy decreases. To be more specific, each spring exerts forces on its connected vertices according to the crossing angle it faces. If the angle is acute, then the spring push the vertices apart. If the angle is obtuse, then the vertices are pulled toward each other. If it is a right angle, no force is applied. The magnitude of the force is $k_{cos} \cos\theta$, where θ ($\theta < 90°$) is the crossing angle between the two edges, and k_{cos} is a constant.

3.2 BIGCROSS Force Direction

During our investigation, we experimented with three different directions in which the cosine force may force crossings toward the right angle.

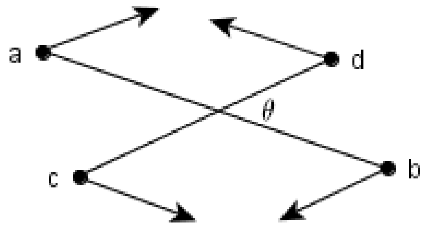


Figure 4: The parallel cosine force

The first one is that the force is applied in the direction of the other crossed edge. To be more specific, suppose that the position of vertex v in an 2D Euclidean space is denoted by $p_v = (x_v, y_v)$, and the distance between vertices u and v is d_{uv}; then given two crossing edges (a, b) and (c, d) as shown in Figure 4, the cosine force on vertex a, which we call *parallel cosine force*, can be denoted as follows:

$$(k_{cos}\cos\theta\frac{x_d - x_c}{d_{cd}}, k_{cos}\cos\theta\frac{y_d - y_c}{d_{cd}}) \tag{3}$$

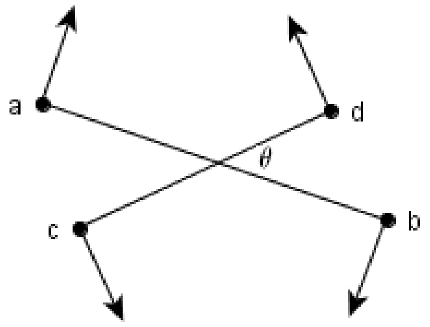


Figure 5: The rotational cosine force

In the second case, the force is applied in the direction orthogonal to the crossed edge. This is a kind of rotational force. Given two crossed edges (a, b) and (c, d) as shown in Figure 5, the cosine force on vertex a is:

$$(-k_{cos}\cos\theta\frac{y_b - y_a}{d_{ab}}, k_{cos}\cos\theta\frac{x_b - x_a}{d_{ab}}) \tag{4}$$

In the third case, the cosine force is divided in two directions. In one direction, the force is applied in an attractive manner and in the other in a repulsive manner. As shown in Figure 6, the attractive component of the cosine force on vertex a is:

$$(k_{cos}\cos\theta\frac{x_d - x_a}{d_{ad}}, k_{cos}\cos\theta\frac{y_d - y_a}{d_{ad}}) \tag{5}$$

and the repulsive component can be denoted as:

$$(k_{cos}\cos\theta\frac{x_a - x_c}{d_{ac}}, k_{cos}\cos\theta\frac{y_a - y_c}{d_{ac}}) \tag{6}$$

Our tests showed that all three produced drawings with larger crossing angles than the classical spring algorithm. However, the best overall result was consistently achieved by the parallel cosine

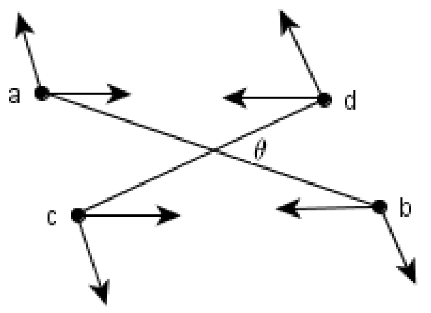


Figure 6: The attractive and repulsive cosine forces

force. For the remainder of this paper, we restrict our attention to the parallel cosine force.

It should be noted that in our implementation, edges that intersect at an endpoint are not considered as crossing.

3.3 The Algorithm

Given a graph $G = (V, E)$, let $V' \subseteq V$ be the set of vertices in which each vertex has at least one incident edge crossing another; the combined force applied on vertex v is:

$$F(v) = \sum_{(u,v)\in E} f_{s,uv} + \sum_{(u,v)\in V\times V} f_{r,uv} + \sum_{(c,v)\in C\times V'} f_{cos,cv} \quad (7)$$

where $f_{s,uv}$ denotes the spring force, $f_{r,uv}$ denotes the repulsive force, $f_{cos,cv}$ denotes the cosine force, and $C \subseteq E \times E$ denotes the set of pairs of edges that cross, in which each pair includes an incident edge of vertex v.

BIGCROSS employs the simple "follow your nose" approach by iteratively calculating forces and updating the positions of vertices accordingly.

4 Experimental Results

Traditionally, in evaluating force-directed methods, the main focus has been on their performance in drawing highly structured graphs, such as planar meshes, trees, hypercubes. However, their usage has gone far beyond those well structured graphs. Nowadays force-directed algorithms are widely used in various application domains to explore real world problems, in which graphs are rarely highly structured. Therefore it is also necessary to investigate how a force-directed algorithm performs with less-structured graphs.

Furthermore, one of known problems with force-directed algorithms is that their performance is not consistent (see, for example, [29]). The quality of the output heavily relies on the combination of input parameters including initial layout and the choice of constants. Fruchterman and Reingold [12] comment that an "algorithm should work reasonably well almost always, without the user having to fiddle with options." Thus we performed experiments based on a set of pre-specified parameters and report our results based on statistical bases.

4.1 Test Data

We have tested on five types of sparse ($|E| \leq 3|V|$) connected graphs listed below. These graphs were either randomly generated based on well accepted models, taken from benchmark data sets, or from those used in previous papers. A more detailed description is as below.

FR graphs: We collected 38 graphs from published papers on force directed methods, mostly taken from Fruchterman and Reingold [12]. These graphs were all structured. Three different initial placements for each graph were produced for testing. In other words, our test for FR graphs was based on 114 cases.

Planar graphs: The planar graphs were taken from the RND/BUP set of GDT testsuites [14]. This graph set originally contains 200 randomly generated undirected planar graphs with size ranging from 10 to 100. For our purpose, only graphs with no more than 50 vertices were used.

Random graphs: We tested on 3 kinds of random graphs [15, 25] generated based on the following three models:

- Erdos-Renyi model [10]
- Watts-Strogatz model [32]
- Eppstein-Wang model [9]

For each model, there were 500 graphs with size ranging from 10 to 50.

4.2 Design

During the testing, the initial layout for each graph was randomly produced confined within a unit square. Then based on the same layout, we ran BIGCROSS and the classical spring algorithm mentioned in Section 2 separately.

We ran pilot studies to determined values for the constants: k_a, k_r, and l were set to 1, and k_{cos} was set to 1. These values work reasonably well together most of the time when an initial placement is confined in a unit square.

Each algorithm stopped once the graph system reached a relatively stable status or iterations reached a pre-specified number. The threshold for the stable status was defined as the time when the maximum movement among all vertices in both x and y direction was no more than 0.0005. The maximum number of iterations was set to 80000. One exception was that for FR graphs, the threshold for maximum movement was set being 0.00001 and the maximum number of iterations was 100000, for best possible final layouts.

Experiments were performed on a 2.4GHz laptop with 2.99GB RAM. The running time and the number of iterations were recorded. The initial and final placements for each graph for both algorithms were measured. The aesthetic criteria we used for graph measurement include:

1. Number of crossings
2. Average size of crossing angles
3. Standard deviation of crossing angle
4. Average edge length
5. Standard deviation of edge length
6. Angular resolution (the smallest angle size between two edges incident to the same vertex)

Statistical analysis of the results according to these measures is described in the following section.

4.3 Results

The results are summarized in Tables 1 and 2. It should be noted that most of the data were not normally distributed, due to the nature of the measurements. Thus the median value, rather than mean, was computed across graphs for accuracy. Accordingly, a nonparametric method called the *Wilcoxcon signed-ranks*[33] test on paired data was used for statistical analysis.

As can be seen from Tables 1 and 2, for FR graphs, on average in terms of median, BIGCROSS increased crossing angle by 4.40 degrees, compared to the classical spring algorithm. It also reduced angle deviation by 1.22 degree, average edge length by 0.01 and edge deviation by 0.02. However, the number of crossings was the same in the two conditions, and angular resolution was decreased

Table 1: Medians of testing measures

	Number of crossings		Angle size (deg.)		Angle deviation (deg.)		Angular resolution (deg.)	
Graph type	*BIGCROSS*	*Classical*	*BIGCROSS*	*Classical*	*BIGCROSS*	*Classical*	*BIGCROSS*	*Classical*
FR	4	4	76.40	72.00	7.53	8.75	28.01	31.09
Planar	12	12	72.55	68.27	11.64	14.63	8.52	9.18
Erdos-Renyi	130	135	66.72	62.52	17.00	18.67	0.79	0.94
Watts-Strogatz	13	12	73.02	67.34	10.81	14.53	8.42	8.85
Eppstein-Wang	218	247	65.27	62.06	17.69	18.85	0.43	0.49

Table 2: Medians of testing measures

	Edge length		Edge deviation		Iterations		Running time (sec.)	
Graph type	*BIGCROSS*	*Classical*	*BIGCROSS*	*Classical*	*BIGCROSS*	*Classical*	*BIGCROSS*	*Classical*
FR	1.70	1.71	0.34	0.36	17596	17056	2.18	0.25
Planar	2.02	2.05	0.48	0.51	6635	7175	2.69	0.73
Erdos-Renyi	1.65	1.97	0.53	0.63	4486	5088	4.87	0.53
Watts-Strogatz	1.87	1.92	0.44	0.47	5887	6521	2.57	0.64
Eppstein-Wang	1.43	1.92	0.53	0.64	4056	4712	6.61	0.50

by 3.08 degrees. Wilcoxcon signed-ranks tests indicated that the differences in average angle size, average edge length and edge deviation were statistically significant with $p < 0.01$.

For planar graphs, BIGCROSS increased crossing angle by 4.28 degrees. It also reduced angle deviation by 2.99 degrees, average edge length by 0.03 and edge deviation by 0.03. However, angular resolution was decreased by 0.66 degree, and the number of crossings was not changed. Wilcoxcon signed-ranks tests indicated that all these differences were statistically significant with $p < 0.01$, except that for angular resolution.

For Erdos-Renyi graphs, BIGCROSS increased average crossing angle by 4.20 degrees. It also reduced angle deviation by 1.67 degrees and the number of crossings by 5. Average edge length was decreased by 0.32, and edge deviation was decreased by 0.08. However, angular resolution was decreased by 0.15 degree. Wilcoxcon signed-ranks tests revealed that all these differences were statistically significant with $p \leq 0.001$, except that for angular resolution.

For Watts-Strogatz graphs, BIGCROSS increased average crossing angle by 5.68 degrees. It also reduced angle deviation by 3.72 degrees. Average edge length was decreased by 0.05, and edge deviation was decreased by 0.03. However, angular resolution was decreased by 0.43 degree and the number of crossings was increased by 1. Wilcoxcon signed-ranks tests revealed that all these differences were statistically significant with $p \leq 0.001$, except that for angular resolution and for the number of crossings.

For Eppstein-Wang graphs, BIGCROSS increased average crossing angle by 3.21 degrees. It also reduced angle deviation by 1.16 and the number of crossings by 29. Average edge length was decreased by 0.49, and edge deviation was decreased by 0.11. However, angular resolution was decreased by 0.06 degree. Wilcoxcon signed-ranks tests revealed that all these differences were statistically significant with $p \leq 0.01$, except that for angular resolution.

Our emphasis in this study was on the quality of outputs rather than the runtime. Our implementation uses relatively naive methods to find edge crossings and decrease energy. Thus it is expensive in terms of running time in comparison to more refined force-directed methods (for example, [11]). Further, the need to compute crossings makes it more expensive than a classical spring method. There are many ways in which the cosine force can be computed more efficiently (note that it is a local force), and we leave this for future study.

5 Examples

From section 4, it is clear that BIGCROSS outperforms the classical spring algorithm. In this section, examples are presented in Figures 7 - 15, all of which are taken from our experiments. Coupling with each resulting drawing, measurement data are provided in the format of (the number of crossings, average crossing angle, angle deviation, average edge length, edge deviation). Where the number of crossings is 0, both average crossing angle and angle deviation are set to 0.

It is important to note that our conclusion that BIGCROSS performs better is based on statistical analysis. Given a set of prespecified constant values, there are cases in which the classical spring algorithm performs better, as shown in Figure 15.

6 Conclusion and Future Work

We have introduced and implemented the cosine force in a force-directed algorithm to increase crossing angles. The experimental results demonstrate that applying cosine force in addition to traditional spring forces in graph drawing not only increases crossing angles, but also surprisingly yields overall more aesthetically pleasing drawings. Our main contributions are:

1. Introduction of the cosine force.
2. Implementation of a new aesthetic criterion - crossing angle - in an algorithm.
3. Introduction of a new force-directed method that produces drawings with some important aesthetics being significantly improved, in comparison to the classical spring algorithm.
4. Statistical evaluation of the algorithm against different types of graphs.

For future work, we note that that it is feasible to speed up BIGCROSS using a fast crossing detection method [1] and/or fast energy minimization methods (for example, FADE [28]). Further, the cosine force can be easily integrated into other force-directed methods, such as Hu-Koren methods to reduce warping [22].

Further, our results indicate that a cosine force may be a good way to increase angular resolution at vertices. Two edges incident to a vertex can be considered as being crossed at the vertex. Therefore, applying cosine force on adjacent edges is a natural extension of what has been presented in this paper. Maximizing vertex angular resolution is particularly interesting since crossing angle competes with angular resolution when two adjacent edges cross the same edge.

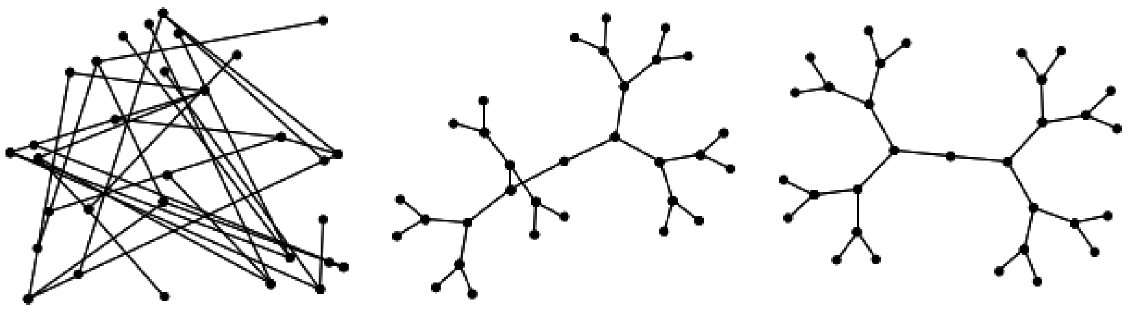

(a) Initial layout (b) Classical: (1, 82.90, 0, 2.08, 0.47) (c) BIGCROSS: (0, 0, 0, 2.07, 0.40)

Figure 7: Tree (graph in Figure 82 from Fruchterman and Reingold [12])

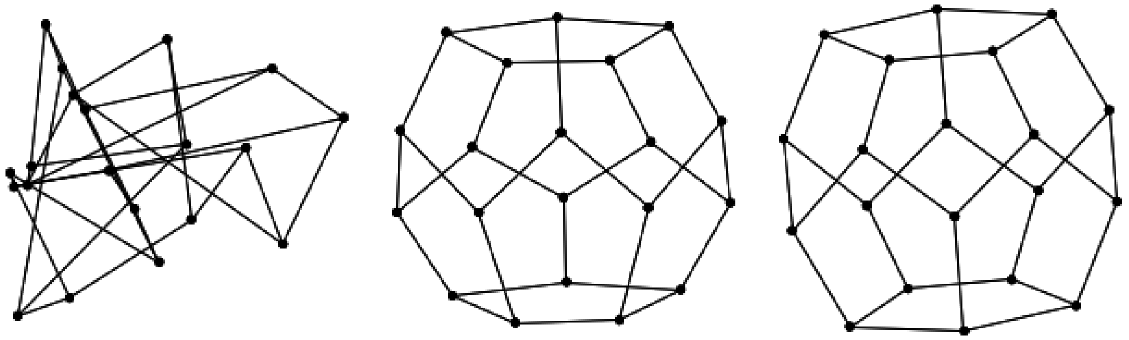

(a) Initial layout (b) Classical: (7, 81.36, 6.32, 1.99, 0.32) (c) BIGCROSS: (6, 86.62, 3.30, 1.99, 0.30)

Figure 8: Dodecahedron (graph in Figure 61 from Fruchterman and Reingold [12])

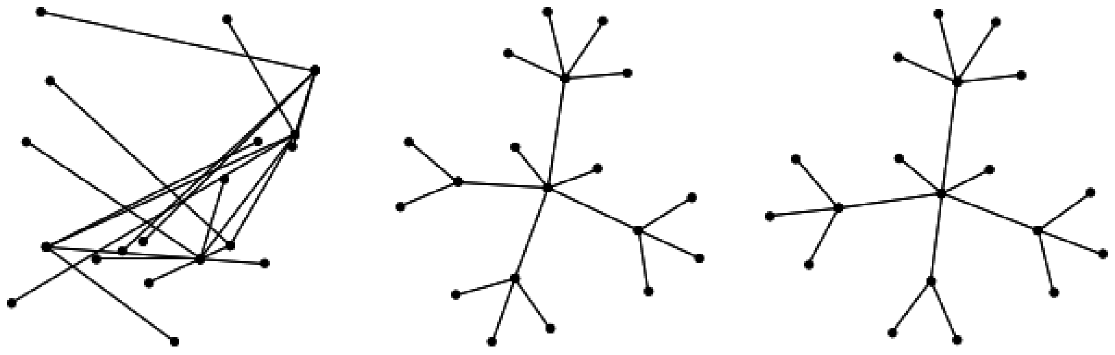
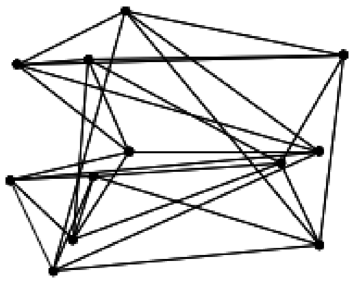
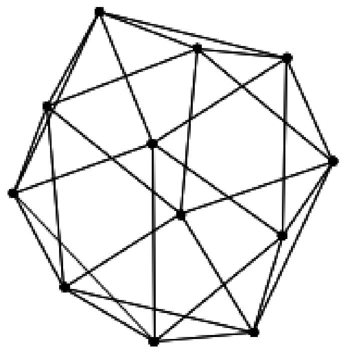

(a) Initial layout (b) Classical: (0, 0, 0, 2.05, 0.47) (c) BIGCROSS: (0, 0, 0, 2.05, 0.46)

Figure 9: Tree (graph in Figure 43 from Fruchterman and Reingold [12])


(a) Initial layout


(b) Classical: (20, 60.22, 16.34, 1.64, 0.41)

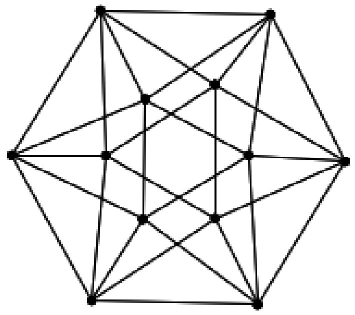
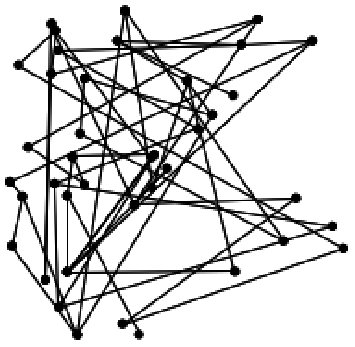
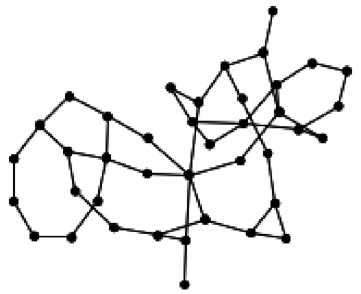

(c) BIGCROSS: (12, 64.49, 4.80, 1.60, 0.29)

Figure 10: Icosahedron (graph in Figure 29 from Fruchterman and Reingold [12])


(a) Initial layout


(b) Classical: (10, 64.36, 11.34, 2.07, 0.45)

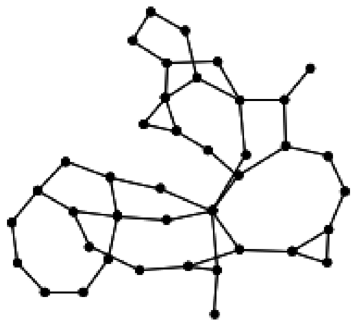
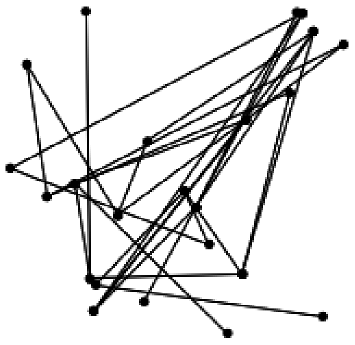
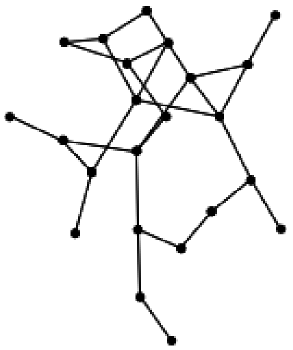

(c) BIGCROSS: (4, 78.36, 2.16, 2.03, 0.34)

Figure 11: A planar graph


(a) Initial layout


(b) Classical: (8, 62.62, 13.15, 1.96, 0.44)

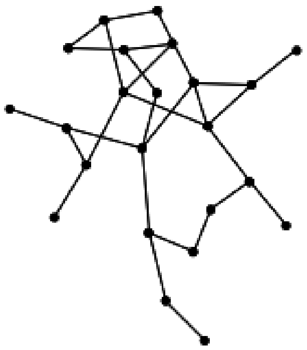
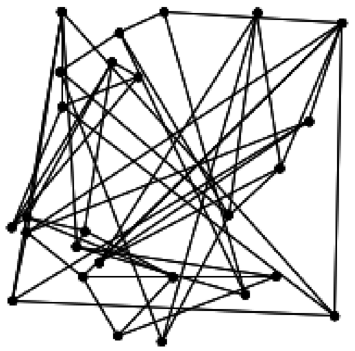
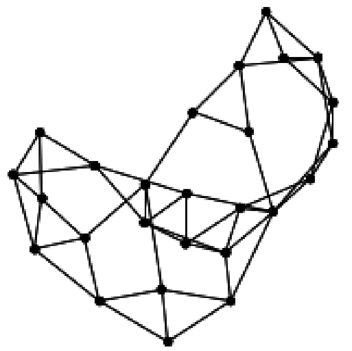

(c) BIGCROSS: (5, 78.06, 4.22, 1.96, 0.43)

Figure 12: A graph of Erdos-Renyi model


(a) Initial layout


(b) Classical: (15, 55.66, 21.73, 1.79, 0.49)

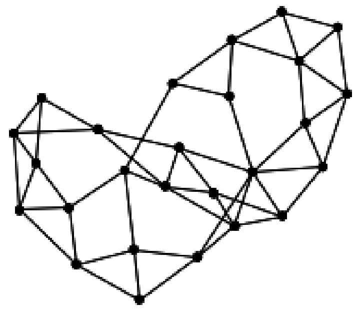
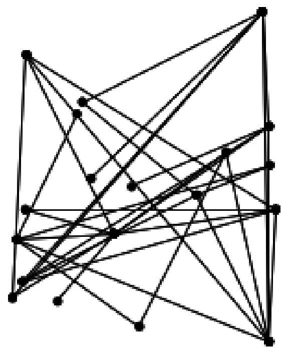
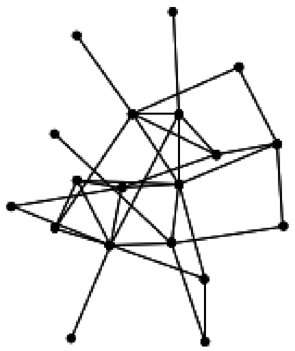

(c) BIGCROSS: (7, 78.89, 7.54, 1.78, 0.41)

Figure 13: A graph of Watts-Strogatz model


(a) Initial layout


(b) Classical: (26, 62.21, 16.15, 1.83, 0.60)

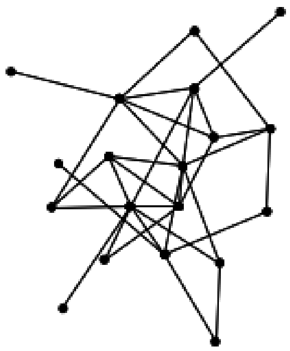
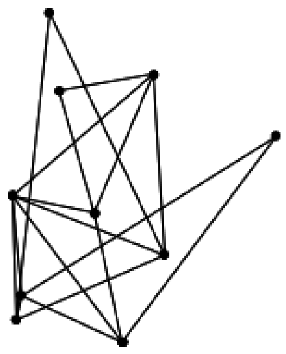
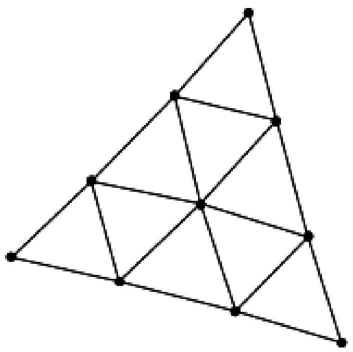

(c) BIGCROSS: (22, 70.57, 15.25, 1.77, 0.51)

Figure 14: A graph of Eppstein-Wang model


(a) Initial layout


(b) Classical: (0, 0, 0, 1.64, 0.06)

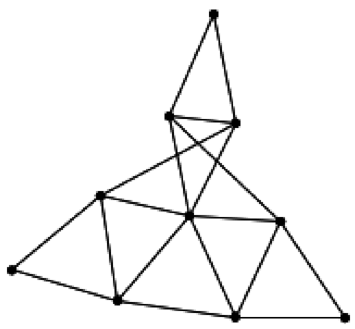

(c) BIGCROSS: (3, 72.69, 0.74, 1.61, 0.28)

Figure 15: A triangulated triangle (graph in Figure 65 from Fruchterman and Reingold [12])

References

[1] Balaban, I.J. (1995) An optimal algorithm for finding segments intersections. In Proc. *the Eleventh Annual Symposium on Computational Geometry* (SCG'95), 211-219.

[2] Brandenburg, F., Himsholt, M. and Rohrer, C. (1996) An experimental comparison of force-directed and randomized graph drawing algorithms. In Proc. *the International Symposium on Graph Drawing* (GD'95). LNCS vol. 1027, Springer-Verlag, 76-87.

[3] Brandes, U. (2001) Drawing on Physical Analogies. In Michael Kaufmann and Dorothea Wagner (Eds.): *Drawing Graphs: Methods and Models*. LNCS Tutorial 2025, Springer-Verlag, 71-86.

[4] Brandes, U. and Wagner, D. (2001) Using Graph Layout to Visualize Train Interconnection Data. *Journal of Graph Algorithms and Applications*, 4(3):135-155.

[5] Didimo, W., Eades, P. and Liotta, G. (2009) Drawing graphs with right angle crossings. In Proc. *the Algorithm and Data Structure Symposium* (WADS 2009), 206-217.

[6] Di Battista, G., Eades, P., Tamassia, R. and Tollis, I. (1998) *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall, Upper Saddle River, New Jersey.

[7] Dunne, C. and Shneiderman, B. (2009) Improving graph drawing readability by incorporating readability metrics: a software tool for network analysts. University of Maryland, Technical Report HCIL-2009-13, May, 2009.

[8] Eades, P. (1984) A Heuristic for Graph Drawing. *Congressus Numerantium*, 42: 149-160.

[9] Eppstein, D. and Wang, J.Y. (2002) A steady state model for graph power laws. In Proc. *2nd International Workshop on Web Dynamics*, May 2002.

[10] Erdos, P. and Renyi, A. (1959) On Random Graphs. *Publicationes Mathematicae*, 6: 290-297.

[11] Frick, A., Ludwig, A. and Mehldau, H. (1995) A fast adaptive layout algorithm for undirected graphs. In proc. *the 2nd Symposium on Graph Drawing*, LNCS vol. 894, Springer-Verlag, 388-403.

[12] Fruchterman, T. and Reingold, E.M. (1991) Graph Drawing by Force-Directed Placement. *Software - Practice and Experience*, 21: 1129-1164.

[13] Gansner, E.R. and North, S.C. (1998) Improved force-directed layouts. In Proc. *the International Symposium on Graph Drawing*, LNCS vol. 1547, Springer-Verlag, 364-373.

[14] GDT (2009) http://www.dia.uniroma3.it/∼gdt/gdt4/test_suite.php

[15] Hayes, B. (2000) Graph Theory in Practice: Part II. *American Scientist*, 88(2): 104. The Scientific Research Society.

[16] Holten, D. and van Wijk, J.J. (2009) Force-directed edge bundling for graph visualization. *Computer Graphics Forum*, 28(3): 983-990.

[17] Hong, S.-H., Merrick, D. and Nascimento, H. (2006) The Metro Map Layout Problem. *Journal of Visual Language and Computing*, 17(3): 203-224.

[18] Huang, M.L., Eades, P., Wang, J. (1998) On-line Animated Visualization of Huge Graphs Using a Modified Spring Algorithm. *Journal of Visual Language and Computing*, 9(6): 623-645.

[19] Huang, W. (2007) Using eye tracking to investigate graph layout effects. In Proc. *Asia-Pacific Symposium on Information Visualisation*, 97-100.

[20] Huang, W., Hong, S.-H. and Eades, P. (2008) Effects of crossing angles. In Proc. *IEEE Pacific Visualization Symposium 2008*, 41-46.

[21] Hu, Y.F. (2005) Efficient and High Quality Force-Directed Graph Drawing. *The Mathematica Journal*, 10: 37-71.

[22] Hu, Y.F. and Koren, Y. (2009) Extending the spring-electrical model to overcome warping effects. In Proc. *IEEE Pacific Visualization Symposium 2009*, 129-136.

[23] Itoh, T., Muelder, C., Ma, K.-L. and Sese, J. (2009) A hybrid space-filling and force-directed layout method for visualizing multiple-category graphs. In Proc. *IEEE Pacific Visualization Symposium 2009*, 121-128.

[24] Kobourov, S.G. and Wampler, K. (2005) Non-Euclidean Spring Embedders. *IEEE Transactions on Visualization and Computer Graphics*, 11(6): 757-767.

[25] Lehmann, K.A. and Kaufmann, M. (2005) Random Graphs, Small-Worlds and Scale-Free Networks. *Peer-to-Peer Systems and Applications 2005*, 57-76.

[26] Lin, C. and Yen, H. (2005) A new force-directed graph drawing method based on edge-edge repulsion. In Proc. *the Ninth international Conference on information Visualisation*, 329-334.

[27] Purchase, H.C., Cohen, R.F. and James, M. (1996) Validating graph drawing aesthetics. In Proc. *the 4th International Symposium on Graph Drawing* (GD'96), Springer, 435-446.

[28] Quigley, A. and Eades, P. (2001) FADE: Graph drawing, clustering, and visual abstraction. In Proc. *the 8th international Symposium on Graph Drawing*, LNCS vol. 1984, Springer-Verlag, 197-210.

[29] Tunkelang, D. (1994) A Practical Approach to Drawing Undirected Graphs. Technical Report. UMI Order Number: CS-94-161., Carnegie Mellon University.

[30] Van Ham, F. and Rogowitz, B. (2008) Perceptual Organization in User-Generated Graph Layouts. *IEEE Transactions on Visualization and Computer Graphics*, 14(6): 1333-1339.

[31] Ware, C., Purchase, H., Colpoys, L. and McGill, M. (2002) Cognitive Measurements of Graph Aesthetics. *Information Visualization*, 1: 103-110.

[32] Watts, D.J. and Strogatz, S. (1998) Collective Dynamics of Small World Networks. *Nature*, 393: 440-442.

[33] Wilcoxon, F. (1945) Individual Comparisons by Ranking Methods. *Biometrics*, 1: 80-83.